Basic Spring Project Steps

1. IntelliJ : Download **community edition** for respective OS
   https://www.jetbrains.com/idea/download/#section=windows

   a. Quickstart maven project : Page 2
   b. WebApp Maven Project : Page 7

2. Eclipse : Download IDE for java and web developers
   https://www.eclipse.org/downloads/packages/release/2022-09/r/eclipse-ide-enterprise-java-and-web-developers

   a. Quickstart maven project : Page 12
   b. WebApp Maven Project : Page 21

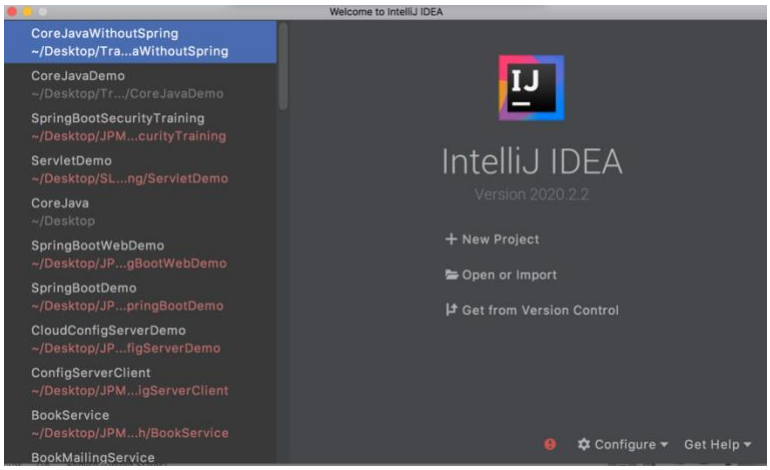
3. STS : Download for respective OS
   https://spring.io/tools

   a. Quickstart maven project : Page 12
   b. WebApp Maven Project : Page 21

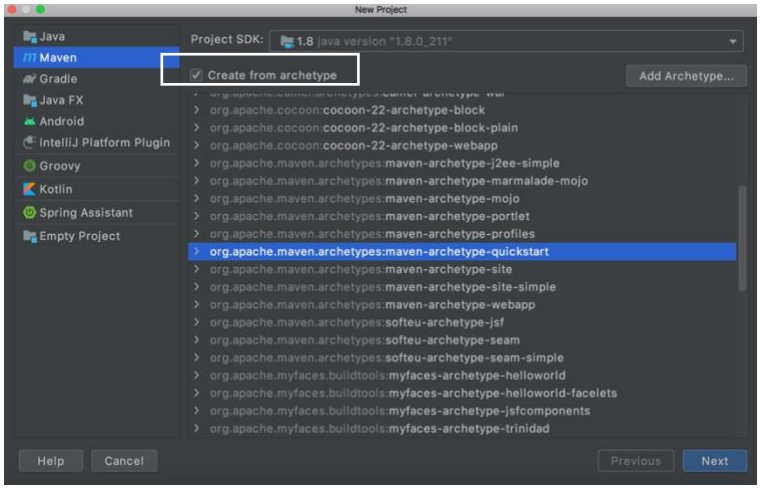
Note: For eclipse and STS the steps are almost the same. Anywhere for the differences will be highlighted

1. **Maven Project: IntelliJ**
    a. QuickStart Maven Project

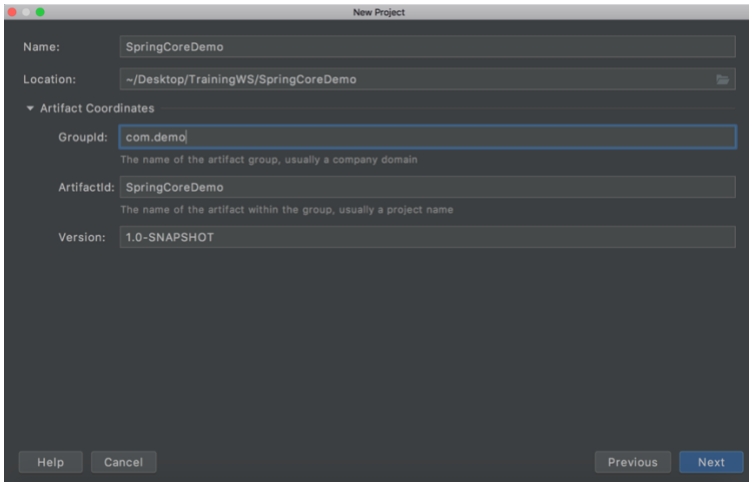
Open IntelliJ and select New Project



Select maven , tick the **checkbox** and select the archetype as highlighted:



Click next and enter the project name. Click on Artifact Coordinates and change the group id. Rest keep the defaults and click next -> next -> Finish



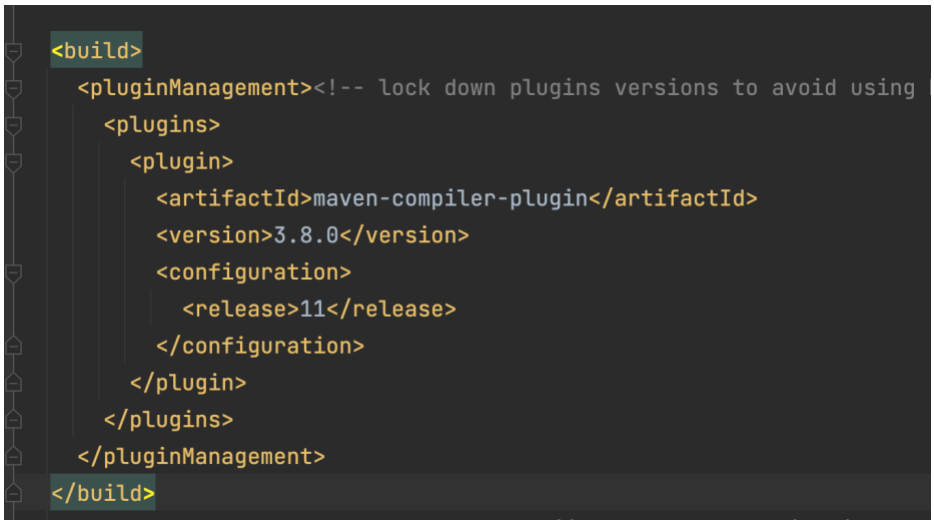
Once the project is created update the java version in pom.xml file as follows:

1. JDK 8 and below : Just update the version with the one installed on your machine as follows

2. For JDK 9 and beyond : Update the version and also update the compiler plugin as follows:

To update the compiler plugin , add release as shown below:

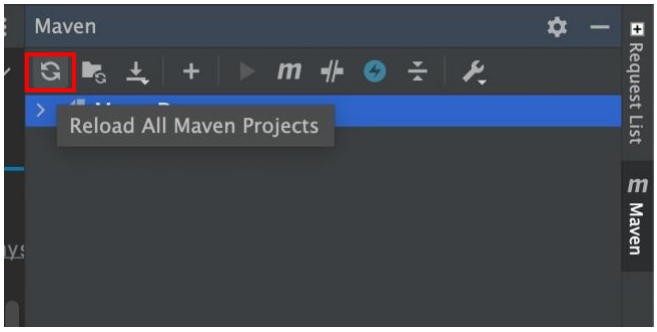
If the above plugin is not there, then add it as shown below within the <build> tag. **The <build> tag will be in the last just before the closing </project> tag**



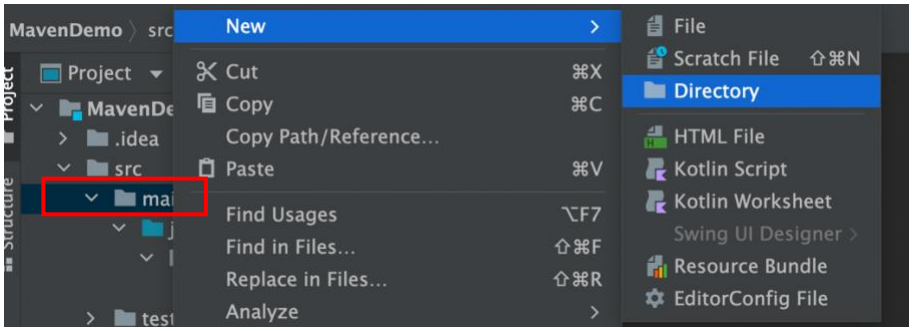
Next step is to add spring dependencies as follows within the <dependencies> tag

```
<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<java.version>11</java.version>
		<spring.version>5.3.23</spring.version>
</properties>
<dependencies>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-context</artifactId>
		<version>${spring.version}</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-tx</artifactId>
		<version>${spring.version}</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-jdbc</artifactId>
		<version>${spring.version}</version>
	</dependency>
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>8.0.28</version>
	</dependency>
	<dependency>
		<groupId>junit</groupId>
		<artifactId>junit</artifactId>
		<version>3.8.1</version>
		<scope>test</scope>
	</dependency>
</dependencies>
```

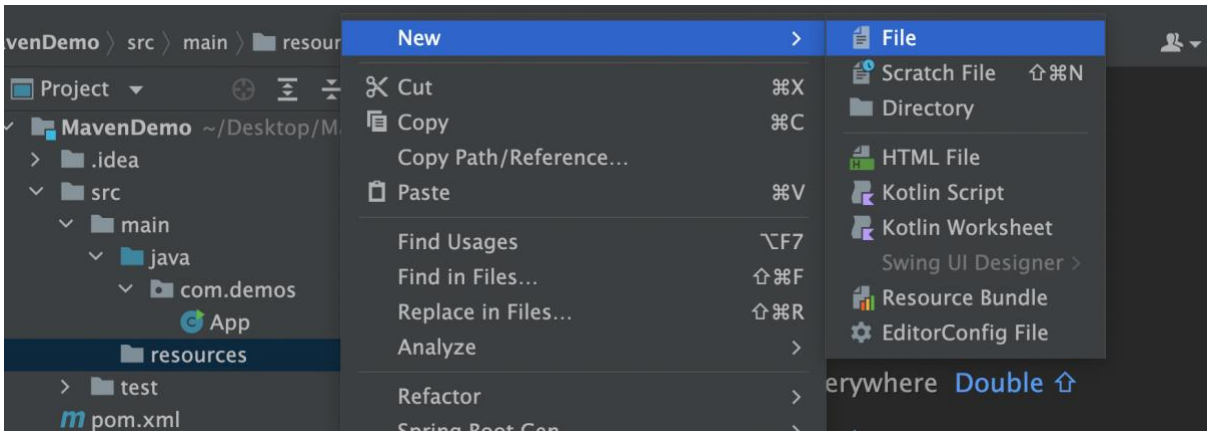
Once dependencies are added, click on Maven tab on the right side of the editor and reload maven project as shown below:



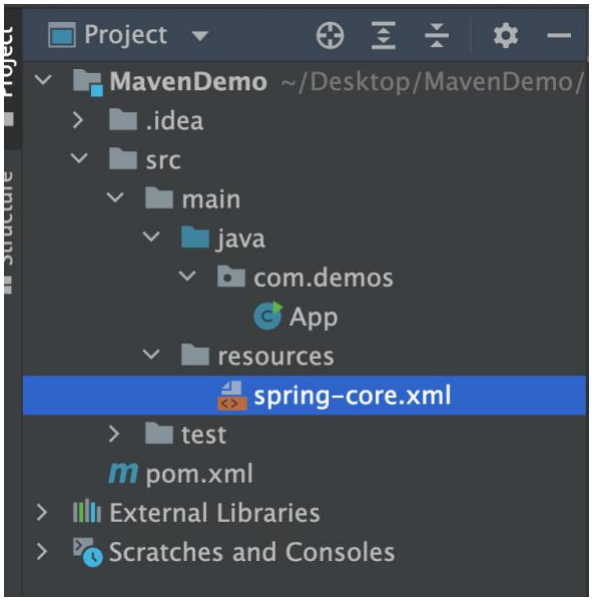
Next create resources folder by right click on main => New => Directory. Give directory name as resources:



Then right click on resources folder => New => File and create a file with the name spring-core.xml



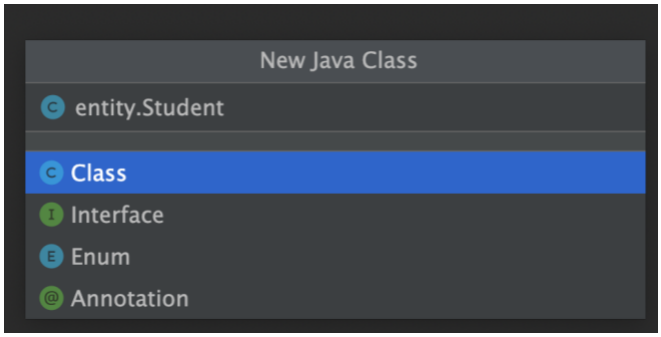
The project structure looks as follows:



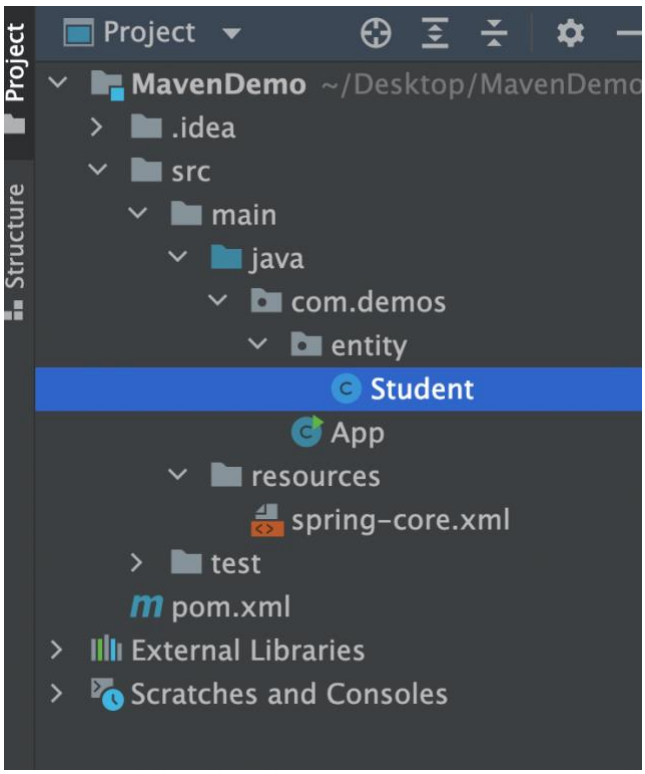
Add the below code in spring-core.xml file:

```
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">

</beans>
```

Create a class Student by right click on com.demos folder and create as shown below:

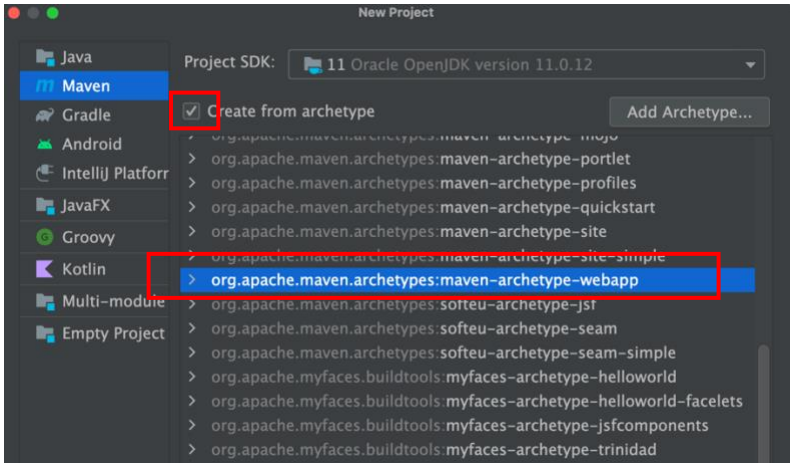


Project structure looks as follows:

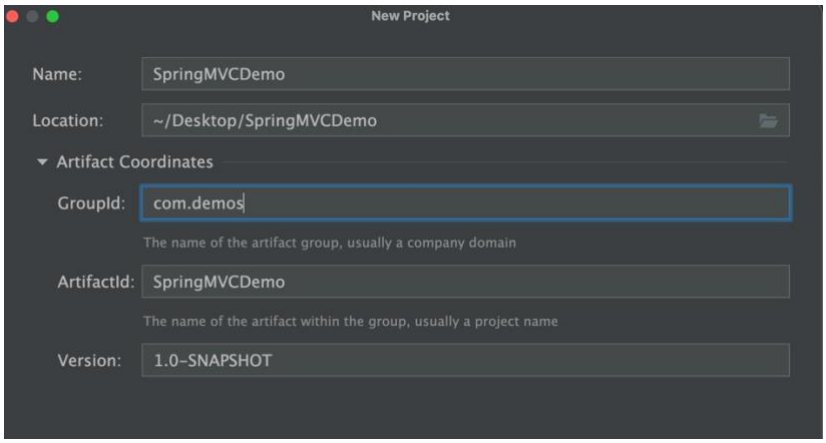


b. WebApp Maven Project

Open IntelliJ and select New Project. Select maven , tick the **checkbox** and select the archetype as highlighted:



Enter the values as shown below, click next -> next -> finish



Update the pom.xml for respective java version as done in previous project
Add the below spring version under <properties> tag

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <java.version>11</java.version>
  <spring.version>5.3.23</spring.version>
</properties>
```

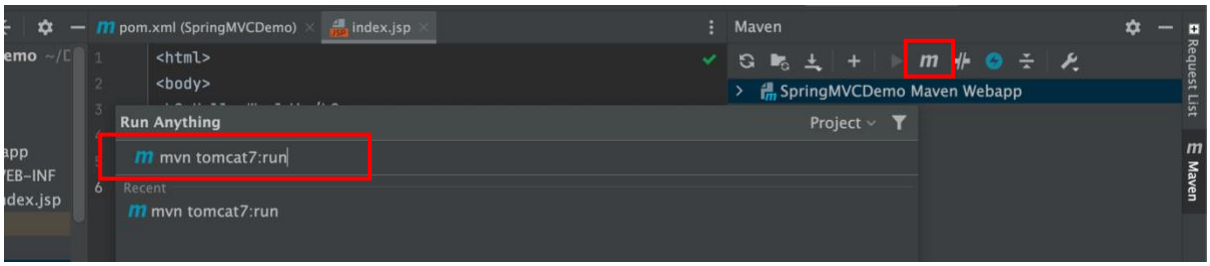
Add the spring dependencies within <dependencies> tag and reload maven clicking on reload button of Maven tab on the right as done in previous project

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${spring.version}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${spring.version}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${spring.version}</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
        <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp.jstl</groupId>
    <artifactId>javax.servlet.jsp.jstl-api</artifactId>
    <version>1.2.1</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
  </dependency>
```

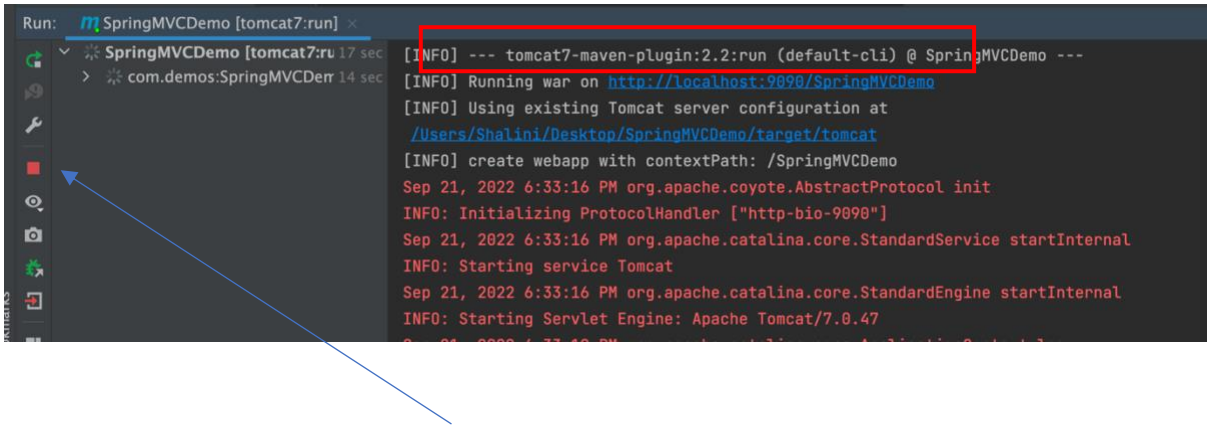
Add tomcat plugin as shown below within the <plugins> tag

```
<plugin>
        <groupId>org.apache.tomcat.maven</groupId>
        <artifactId>tomcat7-maven-plugin</artifactId>
        <version>2.2</version>
        <configuration>
        <port>9090</port>
      </configuration>
</plugin>
```

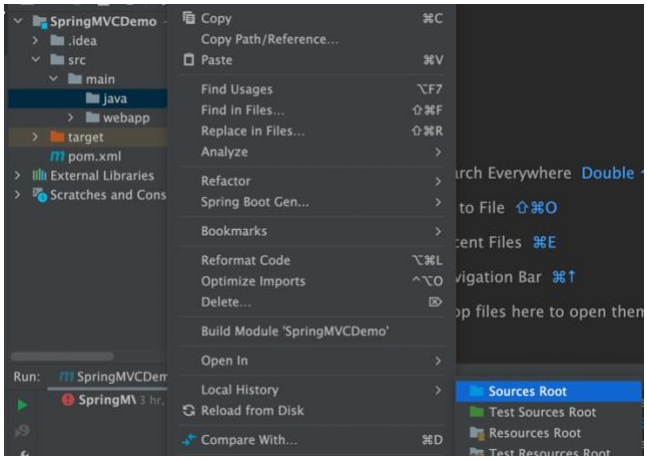
To run web project select m from the maven tab and type in mvc tomcat7:run as shown below and hit enter.

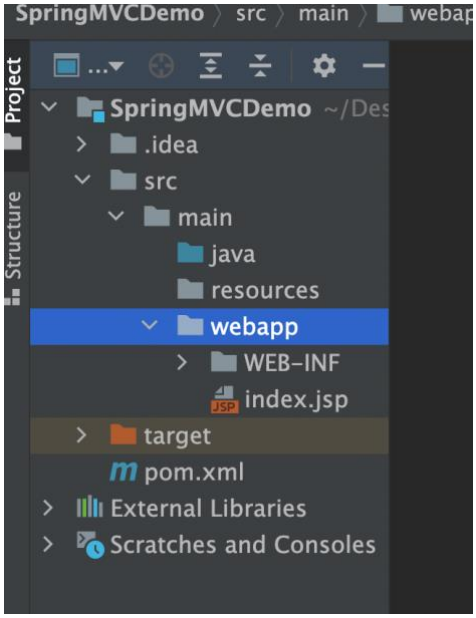


The server should start up and the output should be as shown below: Click on the link as highlighted below and open on any browser of your choice. You should see Hello World on the web page

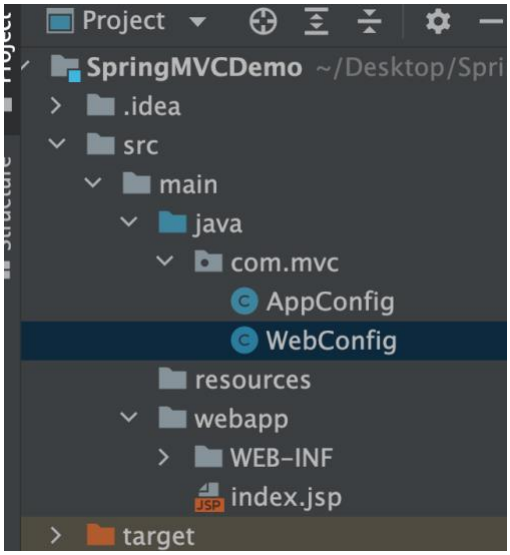


Stop the server by clicking on the red button above. Now right click on main folder => New => directory and give name as java. Mark java as sources root by right click on java folder => Mark Directory as => Sources root. Same way create a folder resources within the mail folder.





Within java folder create 2 classes AppConfig and WebConfig within com.mvc folder as follows:



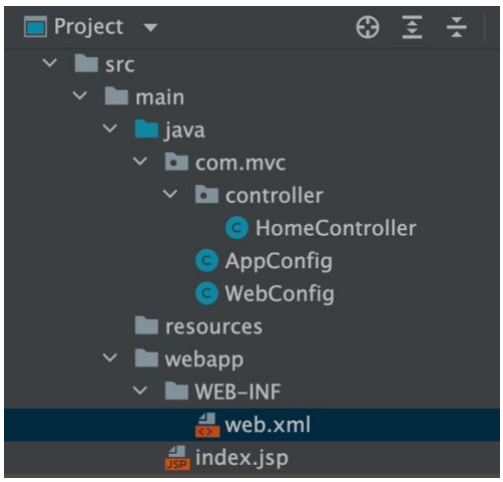
AppConfig.java will have following code:

```
import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class AppConfig extends AbstractAnnotationConfigDispatcherServletInitializer{
	@Override
	protected Class<?>[] getRootConfigClasses() {
		// TODO Auto-generated method stub
		return null;
	}
	@Override
	protected Class<?>[] getServletConfigClasses() {
		// TODO Auto-generated method stub
		return new Class<?>[] {WebConfig.class};
	}
	@Override
	protected String[] getServletMappings() {
		// TODO Auto-generated method stub
		return new String[] {"/"};
	}
}
```

WebConfig.java will have following code:

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
@Configuration
@EnableWebMvc
@ComponentScan
public class WebConfig implements WebMvcConfigurer{
        @Bean
        public InternalResourceViewResolver viewResolver() {
                InternalResourceViewResolver vr = new InternalResourceViewResolver();
                // set location of views.
                vr.setPrefix("/");
                // set the extension of views.
                vr.setSuffix(".jsp");
                return vr; }
}
```

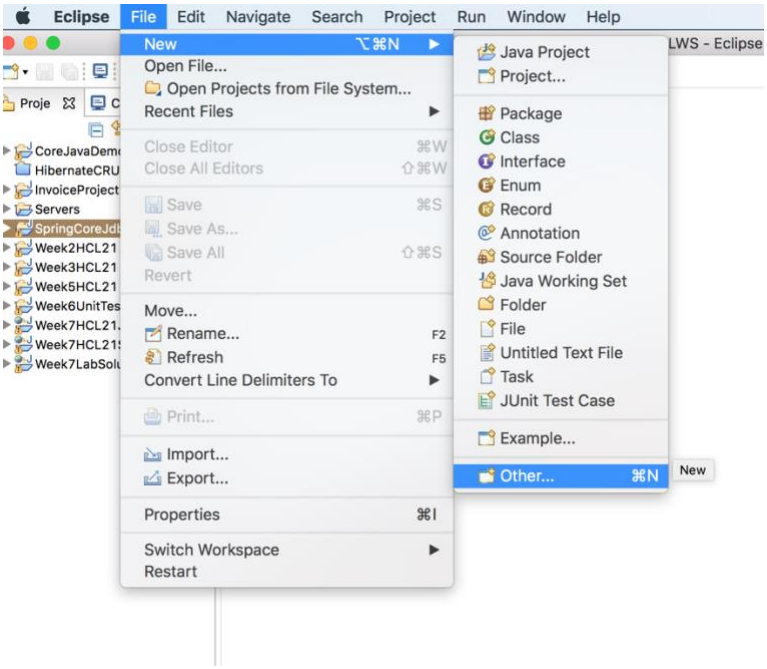
Create a class HomeController within com.mvc.controller directory as follows: Restart the server and access greet on the browser as http://localhost:9090/SpringMVCDemo/greet



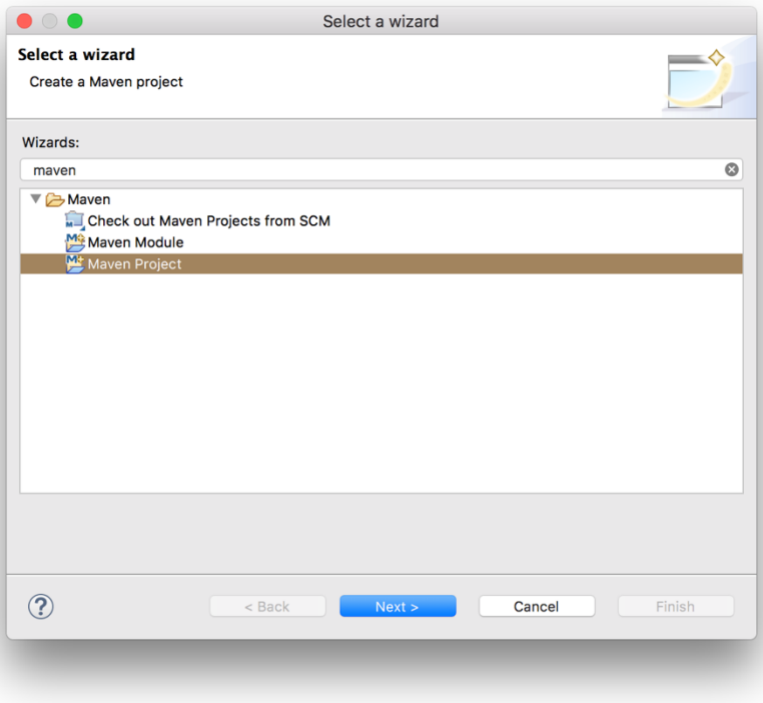

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class HomeController {
   public HomeController(){
      System.out.println("Home controller constructor");
   }
   @RequestMapping("/greet")
   public String greet(){
      System.out.println("greet");
      return "index";
   }
}
```

1. **Maven Project: Eclipse / STS**
    a. QuickStart Maven Project

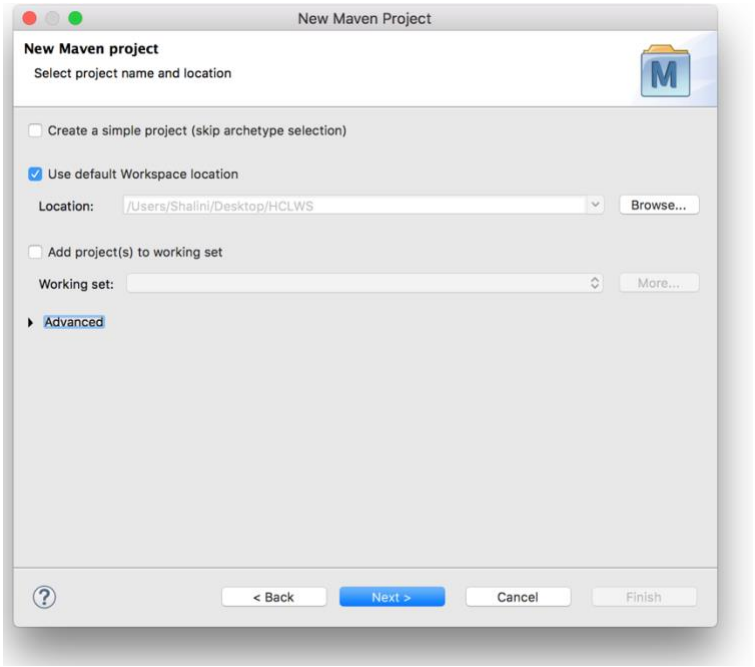
Click on File =>New => Other



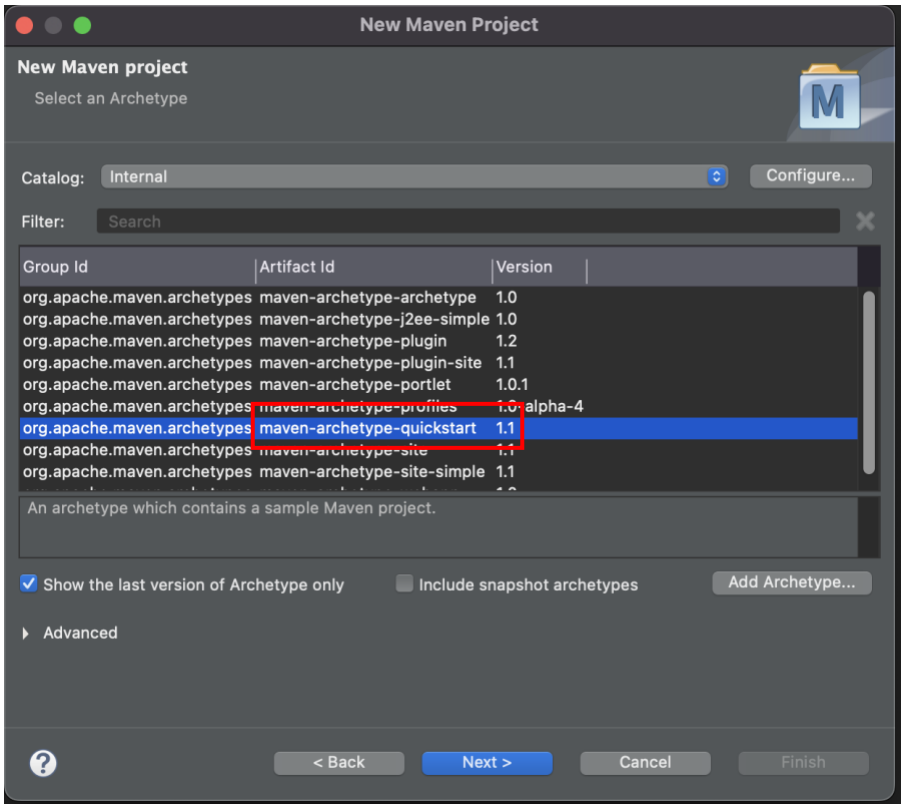
Select Maven Project as follows:



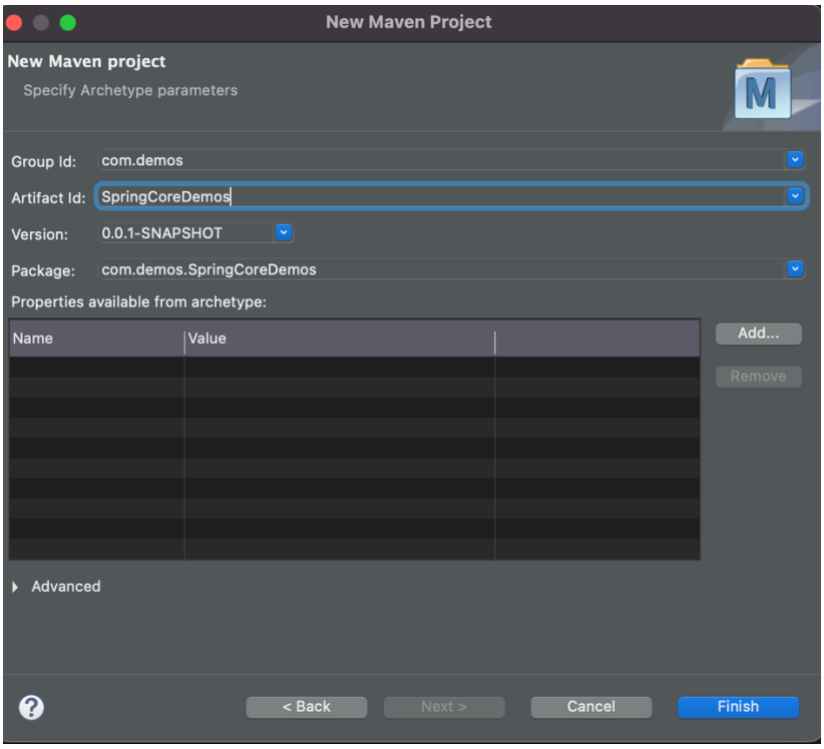
Make sure workspace is your workspace and click Next



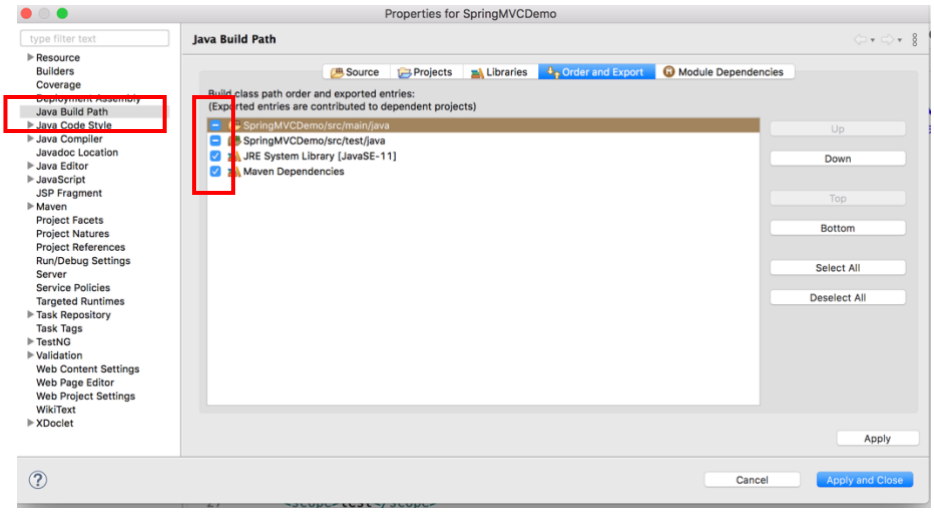
Select the archetype as highlighted below and click next



Enter the group id : com.demos
artifact id : SpringCoreDemo
Click finish



Once the project is created , if you don't see src/main/java then tick the below checkbox after going to Project => Right Click => Properties => Java Build Path on the project explorer



Once the project is created update the java version in pom.xml file as follows:

1. JDK 8 and below : Just update the version with the one installed on your machine as follows

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

2. For JDK 9 and beyond : Update the version and also update the compiler plugin as follows:

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <java.version>11</java.version>
</properties>
```

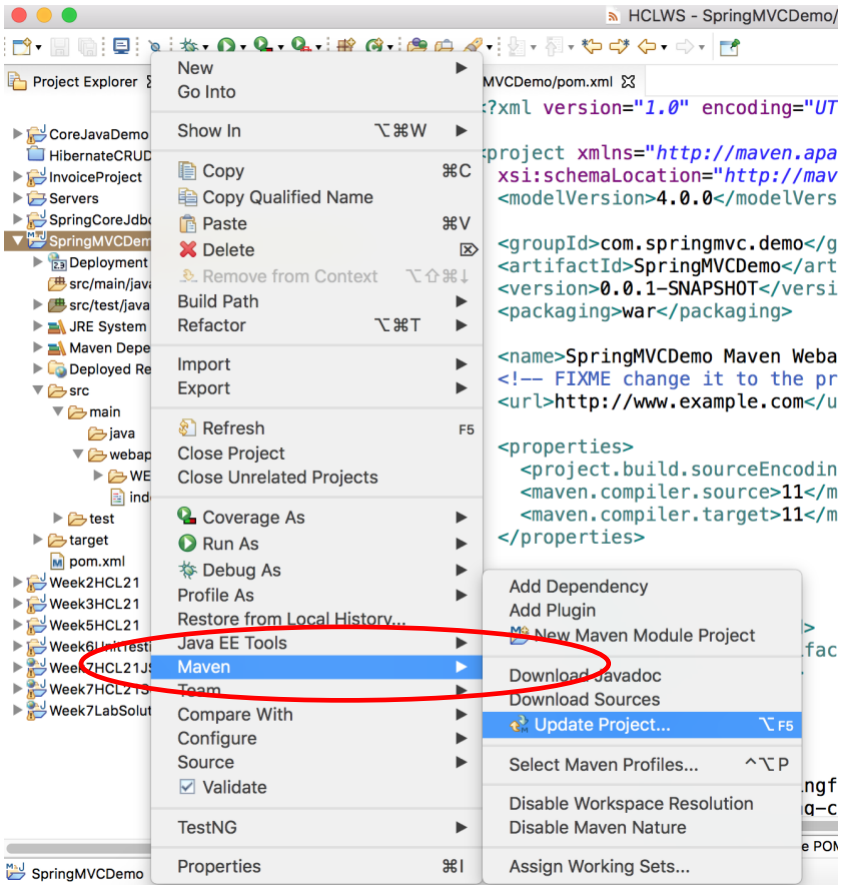
To update the compiler plugin , add release as shown below:

```
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.8.0</version>
  <configuration>
    <release>11</release>
  </configuration>
</plugin>
```

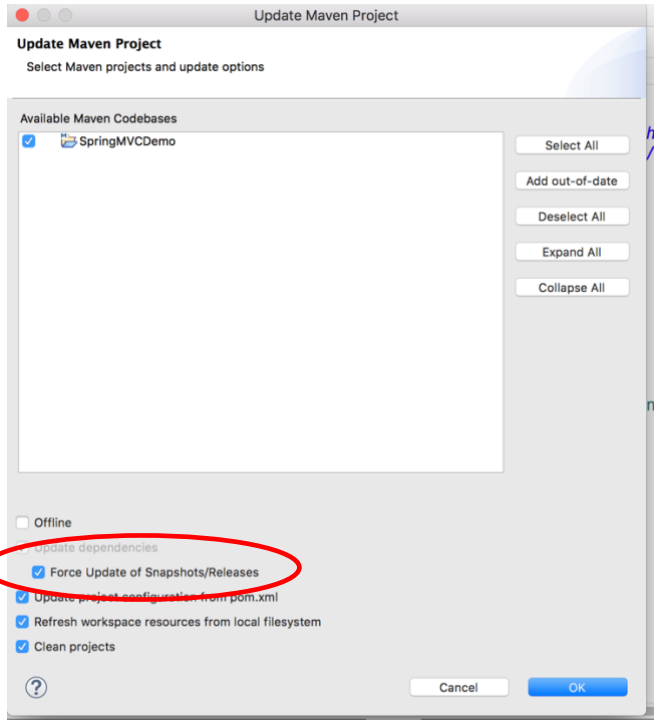
If the above plugin is not there, then add it as shown below within the <build> tag. **The <build> tag will be in the last just before the closing </project> tag**

```
<build>
  <pluginManagement><!-- lock down plugins versions to avoid using
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
        <configuration>
          <release>11</release>
        </configuration>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```

Right click on Project => maven => update project as follows:



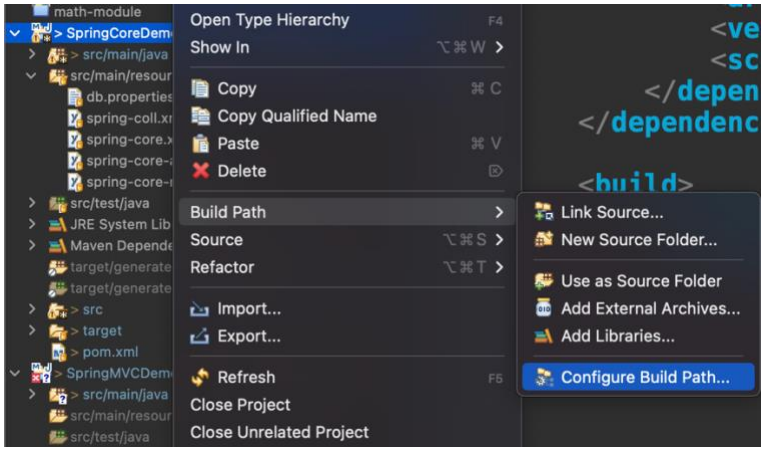
Then click force update and click finish as follows:



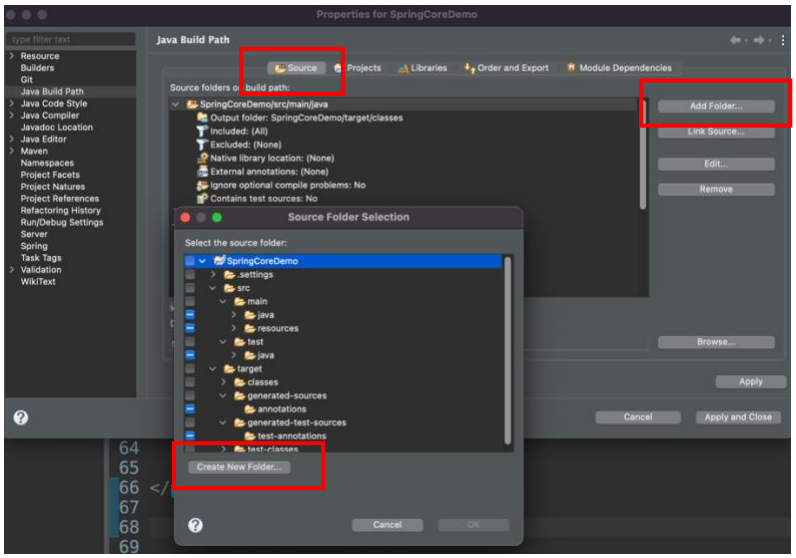
Next step is to add spring dependencies as follows within the <dependencies> tag

```
<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<java.version>11</java.version>
		<spring.version>5.3.23</spring.version>
</properties>
<dependencies>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-context</artifactId>
		<version>${spring.version}</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-tx</artifactId>
		<version>${spring.version}</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-jdbc</artifactId>
		<version>${spring.version}</version>
	</dependency>
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>8.0.28</version>
	</dependency>
	<dependency>
		<groupId>junit</groupId>
		<artifactId>junit</artifactId>
		<version>3.8.1</version>
		<scope>test</scope>
	</dependency>
</dependencies>
```

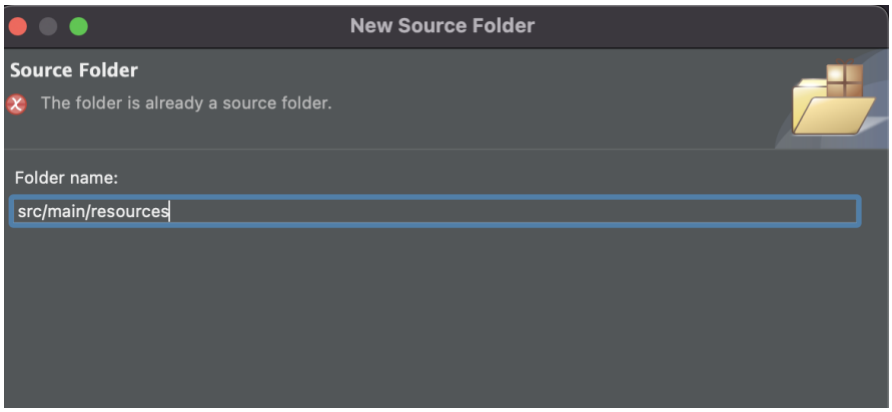
Create src/main/resources folder by Right click on Project => Build Path => Configure Build Path.



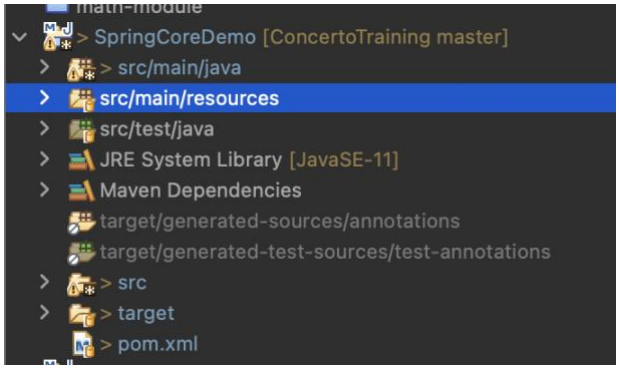
Click Sources tab, then click Add Folder and say Create New Folder



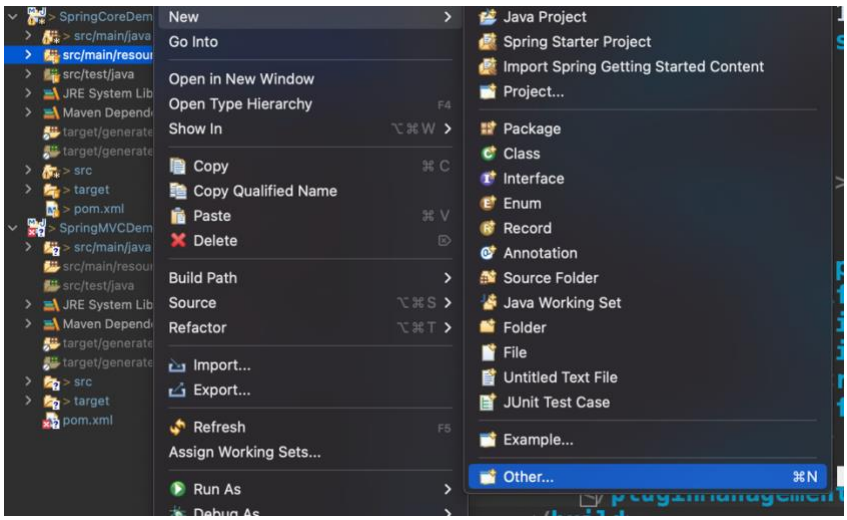
Give the folder name as follows:

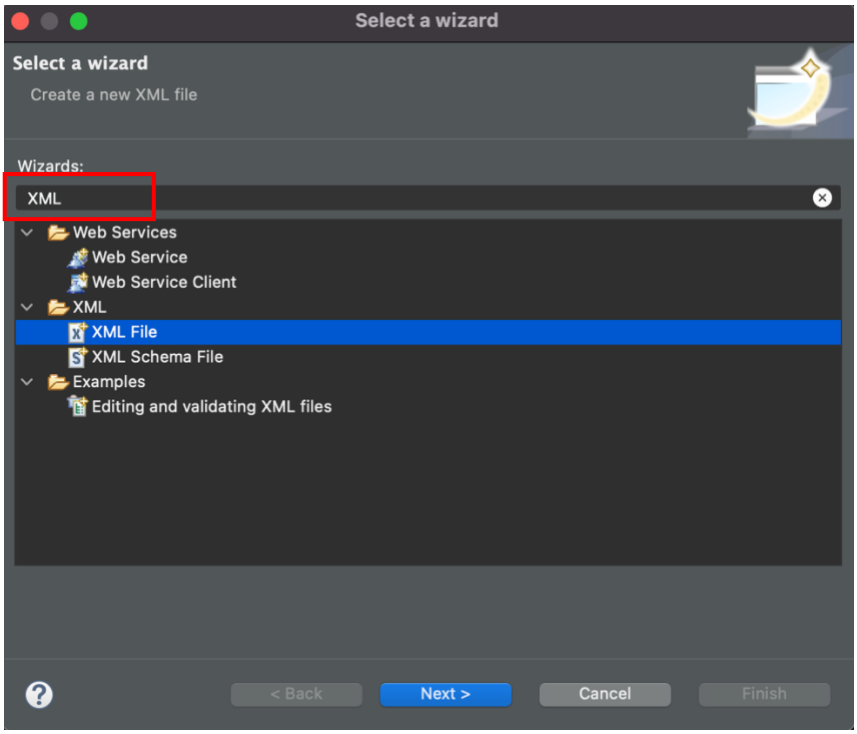


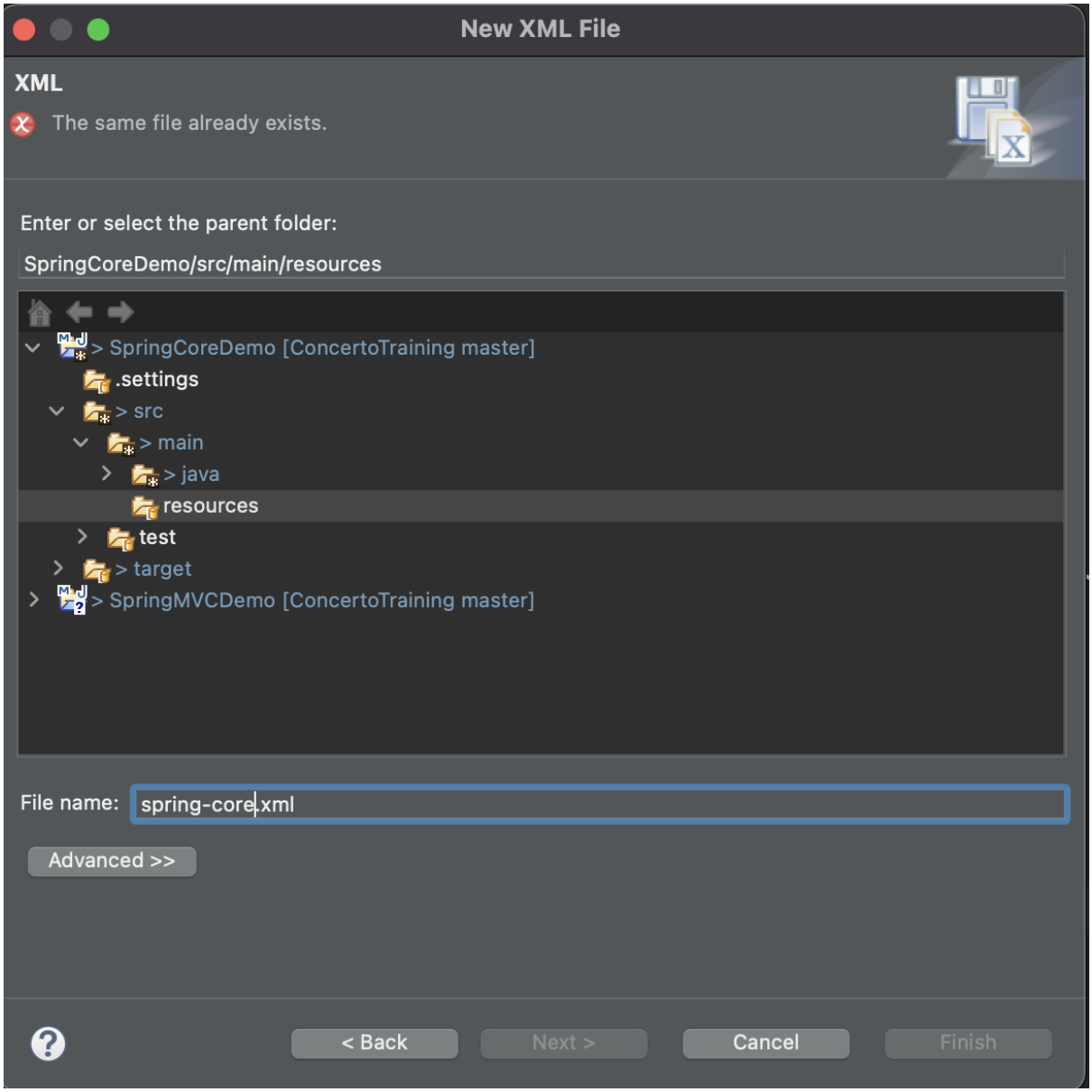
The project structure looks as follows:



Within resources folder, right click => New => Other => XML File and type in the name spring-core.xml and click Finish







Add below code in spring-core.xml file

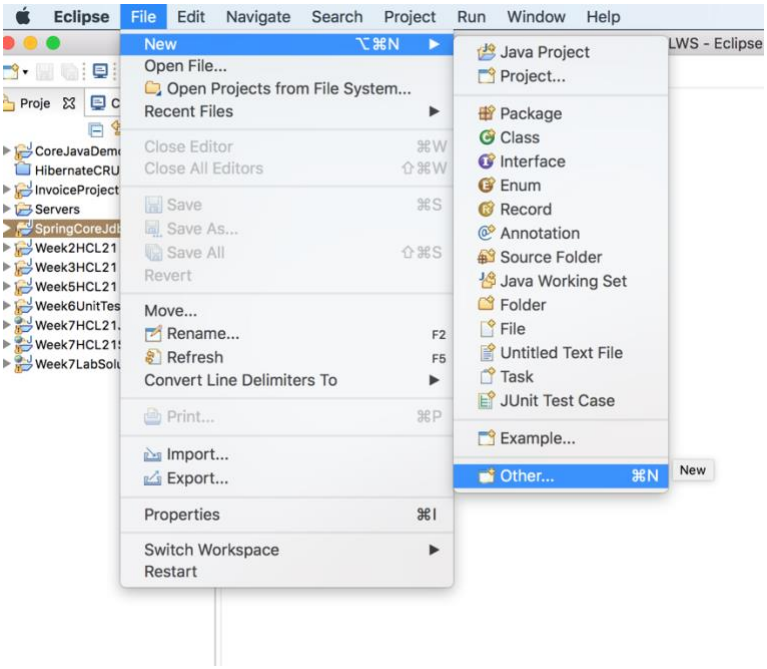
```
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">

</beans>
```

Create a class Student by right click on com.demos folder and create within **com.demos.entity** folder

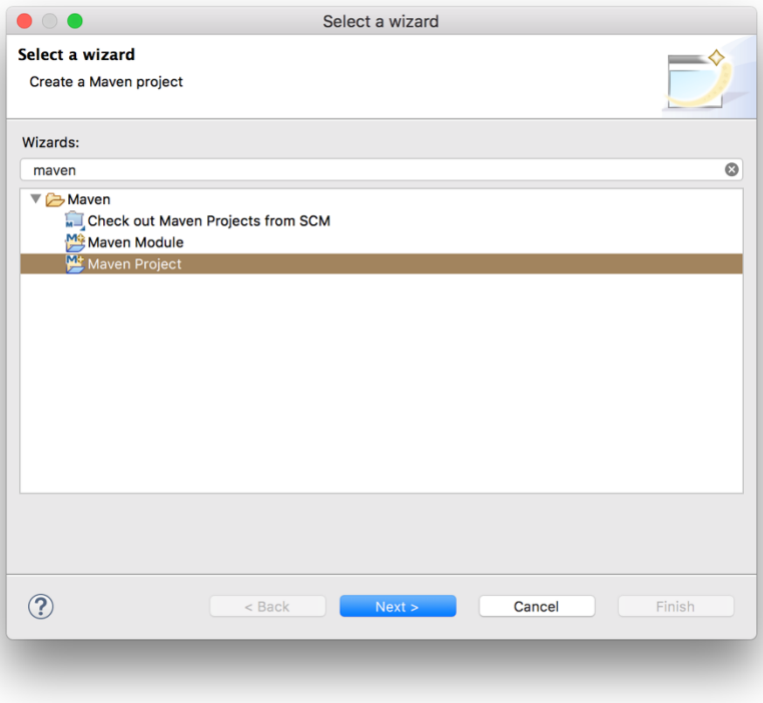
2. Maven Project: Eclipse / STS

b. WebApp Maven Project

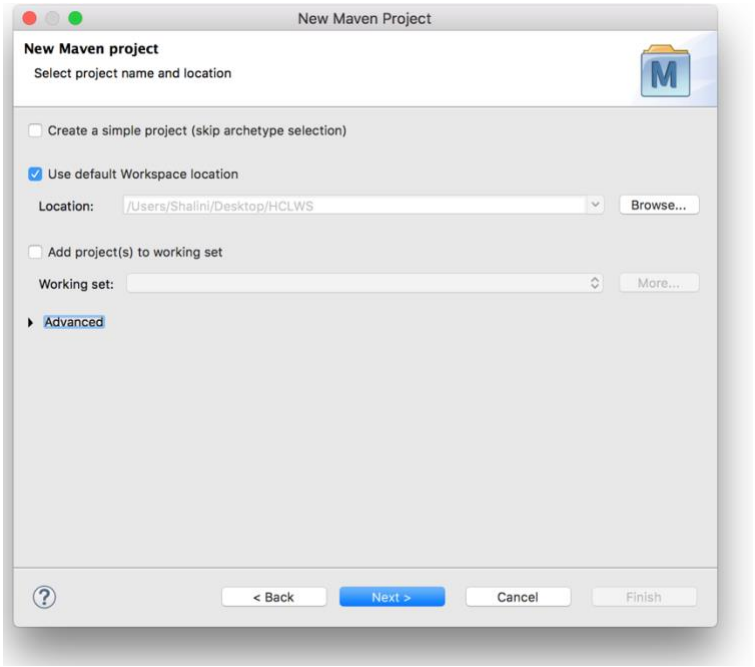
Click on File =>New => Other



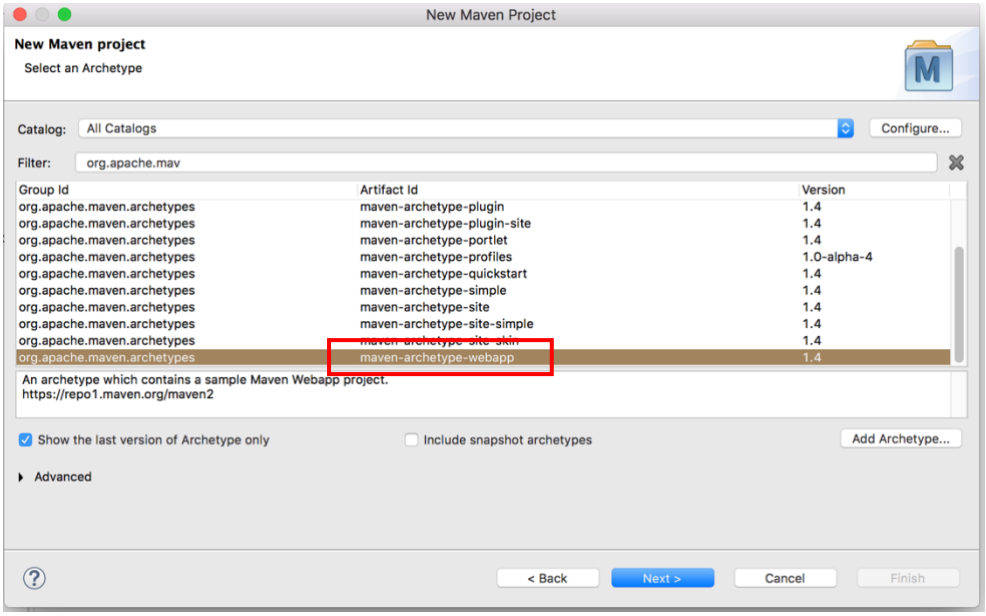
Select Maven Project as follows:



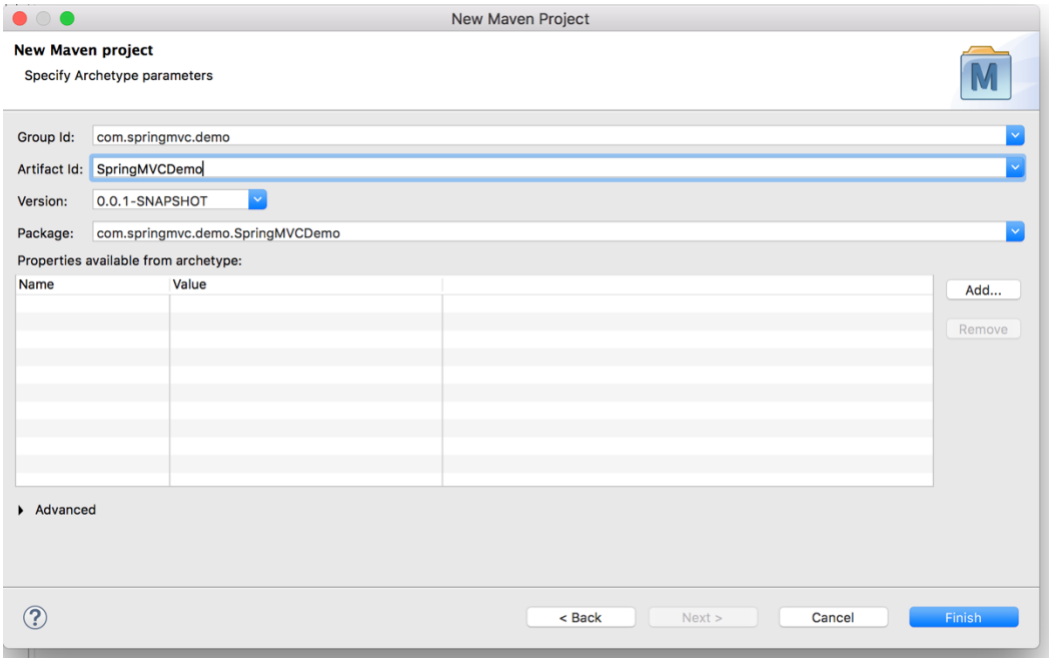
Make sure workspace is your workspace and click Next



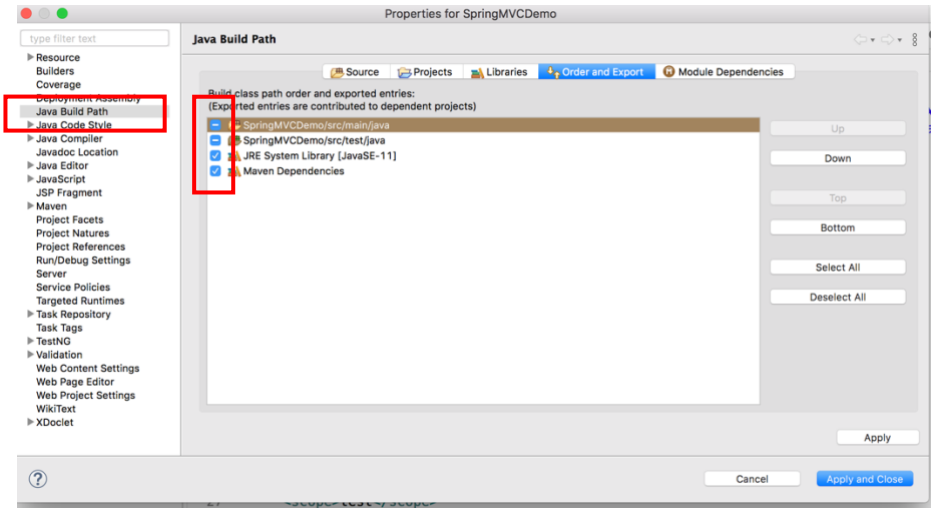
Select the archetype as highlighted below and click next



Enter the group id : com.springmvc.demo
artifact id : SpringMvcDemo



Once the project is created , if you don't see src/main/java then tick the below checkbox after going to Project => Right Click => Properties => Java Build Path on the project explorer



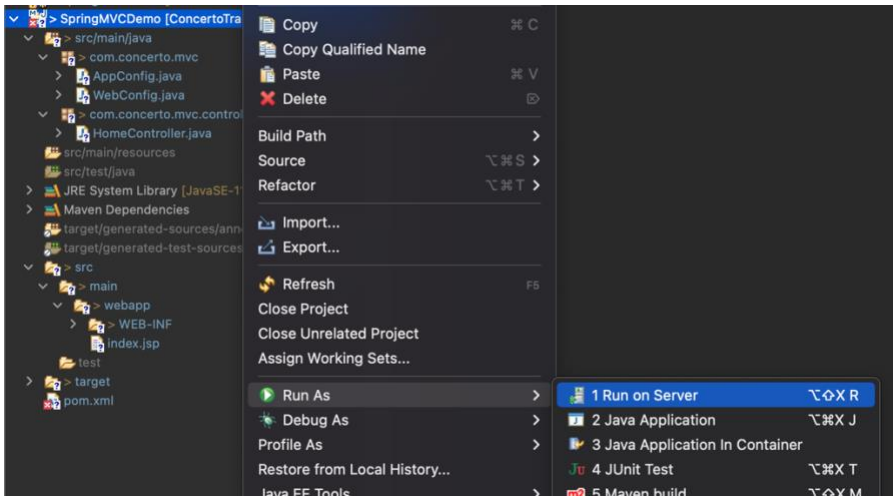
Do update the java version in pom.xml file with respect to java version on your system as explained in the previous project. Also update the Maven project and force update snapshot.

Add the following dependencies in pom.xml file for spring mvc project.

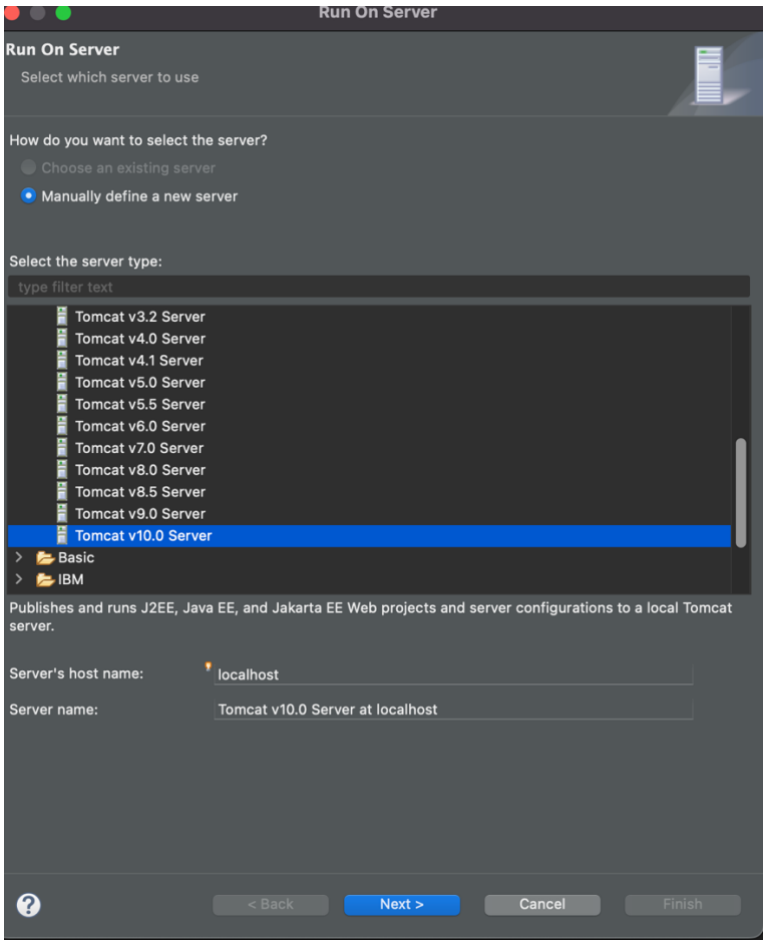
```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <java.version>11</java.version>
  <spring.version>5.3.23</spring.version>
</properties>
```

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${spring.version}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${spring.version}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${spring.version}</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp.jstl</groupId>
    <artifactId>javax.servlet.jsp.jstl-api</artifactId>
    <version>1.2.1</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.1</version>
  </dependency>
```

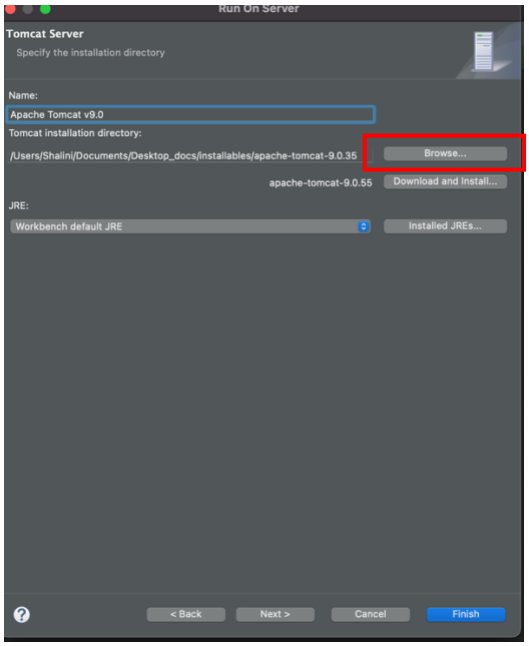
Next right click on project => Run As => Run on Server



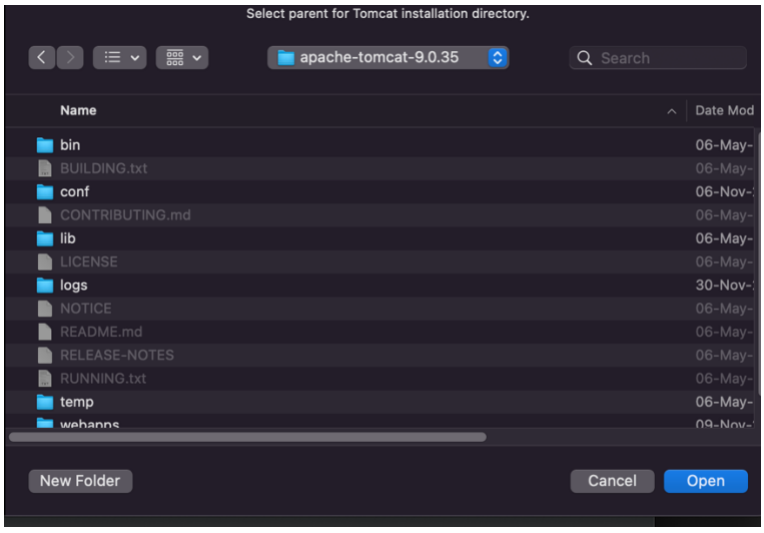
=> Select Apache tomcat 9 or tomcat 10 as per the version you have downloaded on your machine.



Click on Browse

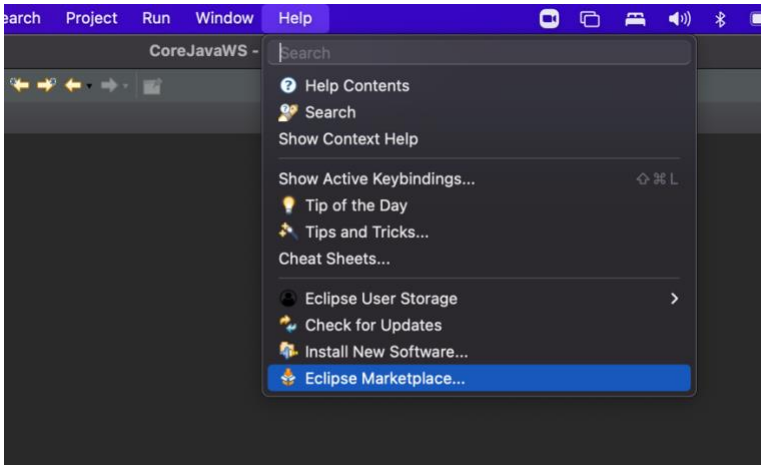


Provide the path to your tomcat folder which is unzipped and downloaded, click Open and then click Finish. The server should start and the project will be available at http://localhost:8080/SpringMVCDemo/ . Hello World should be displayed.

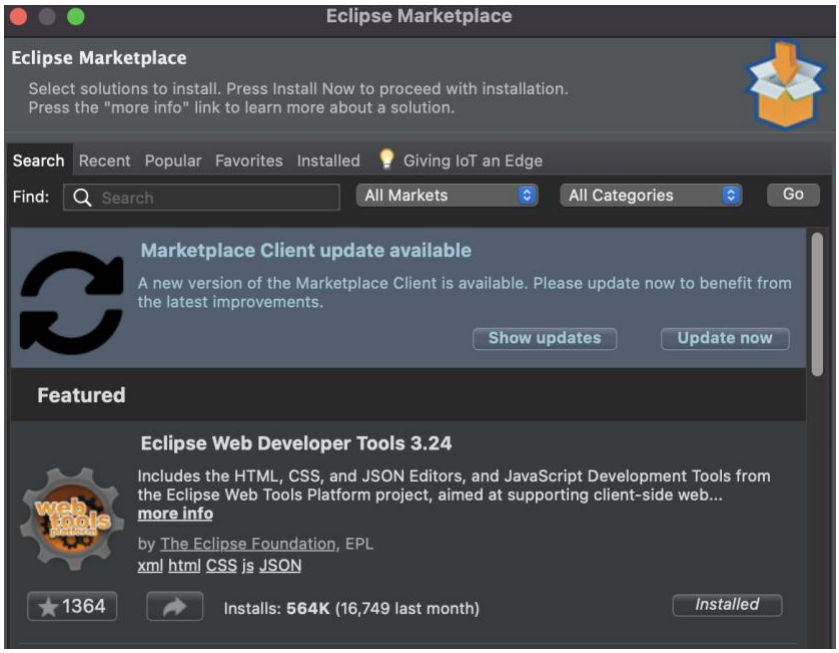


FOR STS user, need to add java EE plugin as follows: [If Run on server is not available]

Click on Help => Eclipse Marketplace



Search for Eclipse Web Developer tools and click on Install and follow the steps of UI

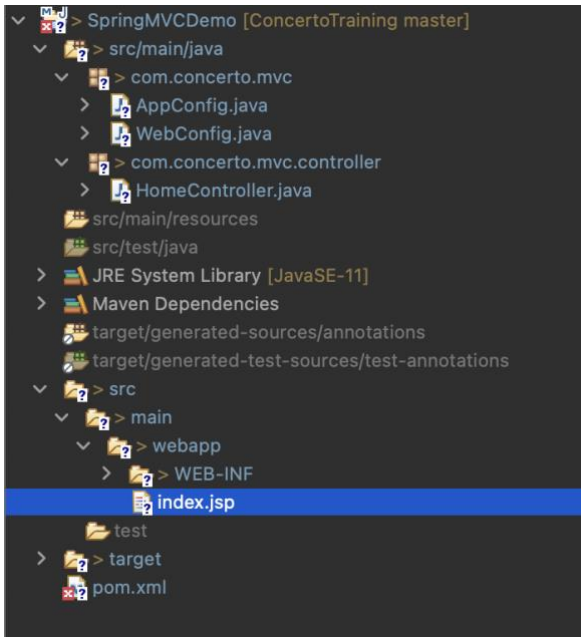


Create 2 classes AppConfig and WebConfig within com.springmvc.demo and add the below code :

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
@Configuration
@EnableWebMvc
@ComponentScan
public class WebConfig implements WebMvcConfigurer{
	@Bean
	public InternalResourceViewResolver viewResolver() {
		InternalResourceViewResolver vr = new InternalResourceViewResolver();
		// set location of views.
		vr.setPrefix("/");
		// set the extension of views.
		vr.setSuffix(".jsp");
		return vr; }
}
```

```
import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class AppConfig extends AbstractAnnotationConfigDispatcherServletInitializer{
	@Override
	protected Class<?>[] getRootConfigClasses() {
		// TODO Auto-generated method stub
		return null;
	}
	@Override
	protected Class<?>[] getServletConfigClasses() {
		// TODO Auto-generated method stub
		return new Class<?>[] {WebConfig.class};
	}
	@Override
	protected String[] getServletMappings() {
		// TODO Auto-generated method stub
		return new String[] {"/"};
	}
}
```

Create a class HomeController within com.springmvc.controller package as follows:



In HomeController.java :

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class HomeController {
   public HomeController(){
      System.out.println("Home controller constructor");
   }
   @RequestMapping("/greet")
   public String greet(){
      System.out.println("greet");
      return "index";
   }
}
```

Rerun the project as Run As => Run On server and restart the server.
Access the greet on browser as follows:

http://localhost:8080/SpringMVCDemo/greet